//在同一刻一个动作只能有一个执行者 ,要达到此效果需要sp->runAction(action->clone():对动作进行克隆 action为两个精灵要执行的动作

//cocos2d-x 生命周期 ，当一个场景进入视线的时候调用OnEnter->添加精灵、开启时间调度、开启触摸

//当一个场景消失在视线的时候，调用OnExit->内存操作、关闭时间调度、关闭触摸

//(如何优化场景跳转时的内存峰值)

//当一个场景跳转到另一个场景（A场景->（跳转动画）B场景）

//先调用A场景的onEnerTransitionDidFinish()->B场景的init->再->再调用A场景的OnExit->调用B场景的OnEnter

创建帧动画的步骤

1. 获取缓存单例，然后再缓存中加载帧

auto cache=SpriteFrameCache::getInstance();

cache->addSpriteFramesWithFile(plist文件);

1. 创建Animation对象

auto animation=Animation::create();

2.1查看plist文件，获取帧的数量count，然后向animation中添加帧

for (int i=0; i<count; i++) {

auto fN=StringUtils::format("%d.png",i+1);

auto aF=cache->getSpriteFrameByName(fN);

animation->addSpriteFrame(aF);

}

2.2设置Animation中每帧的播放时间

animation->setDelayPerUnit(2.0/count);

1. run

创建精灵

auto sp=Sprite::createWithSpriteFrame(cache->getSpriteFrameByName("s\_1.png"));

sp->setPosition(Vec2(480,400));

this->addChild(sp);

3.1 创建播放容器Animate

auto animate=Animate::create(animation);

sp->runAction(传animate);

CocosStudio 下的资源路径在其工程的res文件夹下，那么在res下的文件（非文件夹）需要放入到游戏项目的Resources下，在res下的文件夹需要以真实的路径放到游戏项目的Resources下。

Json解析

1. 拷贝文件到可读写路径下

a获取本地文件（只读的）中的内容（因为即将要把该内容拷贝到可读可写路径下的文件中）

FileUtils::getInstance()->getStringFromFile(本地的文件名);

b 获取可读写路径

//获取在对应设备的可读写路径

auto writablep=FileUtils::getInstance()->getWritablePath();

//+=只能创建文件，不能创建文件夹

writablep+="data.json";

c 判断文件是否存在

if(!FileUtils::getInstance()->isFileExist(writableP)){

//不存在

//c1.

FILE \*file =fopen(writableP.c\_str(), "w");//w 写 打开（可读写路径下的文件），如若不存在就创建一个

if (file) {//文件内容拷贝

fputs(fileData.c\_str(), file);//写入文件

}

fclose(file);//关闭文件

}

1. 解析

a.获取可读写路径下某个文件

auto writableP = FileUtils::getInstance()->getWritablePath();

writableP += 要解析的文件名;

b.获取可读写路径下某个文件中的内容

auto fileData= FileUtils::getInstance()->getStringFromFile(writableP.c\_str());

c.创建工厂Document

rapidjson::Document doc;

//c1.开始解析，解析的东西放到工厂doc中

doc.Parse<kParseDefaultFlags>(fileData.c\_str());

//c2.判断解析是否出错

doc.HasParseError()//返回值为BOOL值，true为有错

d.增删查改的操作

//查HasMenber

if(doc.HasMenber(需要查的key)){

log(“%x”,doc[key].getxxx);}

// 删(删除的是doc中的内容)

doc.RemoveMember("levels");

// 改

doc["username"].SetString("王八");

// 增

//增加value类型为数字

doc.AddMember("test",123,doc.GetAllocator());

//增加value类型为字符串

doc.AddMember("test1","eqweqw",doc.GetAllocator());

//增加value类型为数组

rapidjson::Value v(kArrayType);

v.PushBack(123,doc.GetAllocator());

v.PushBack(1234,doc.GetAllocator());

doc.AddMember("array",v,doc.GetAllocator());

// rapidjson::Value v(kArrayType);

// v.PushBack(123,doc.GetAllocator());

// doc.AddMember("test",v,doc.GetAllocator());

// rapidjson::Value &array = doc["test"];

//这里要说明下getxxx有以下几种类型

简单数据类型、字符串、对象（字典类型）、数组

// 将新的文件内容保存起来

StringBuffer buffer;//相当于包装好的json内容

Writer<StringBuffer> write(buffer);//相当于包装纸

doc.Accept(write);//让doc接受写的命名

//将新的内容保存到文件中去

FILE \*file =fopen(writableP.c\_str(), "w");

if (file) {

fputs(buffer.GetString(), file);

}

fclose(file);

}

打包

open .bash\_profile

第一种 ：使用build\_native.py进行编译打包

1 .环境搭建

在.bash\_profile 中添加以下几行

export NDK\_ROOT=/Users/mac/Documents/NDK/android-ndk-r9d

export PATH=$PATH:$NDK\_ROOT

export SDK\_ROOT=/Users/mac/Documents/NDK/android-sdk-macosx

export PATH=$PATH:$SDK\_ROOT

export ANT\_ROOT=/Users/mac/Documents/NDK/apache-ant-1.9.3/bin

export PATH=$PATH:$ANT\_ROOT

分别是NDK、SDK、ANT的路径，其中ANT需要配置到bin目录

2.修改Android.mk文件

目的是将自己所写的代码文件进行编译

3.在终端执行build\_native.py

4.导入到eclipse中，run as 。。。。。可以在安卓设备中运行,如何进行调试？

在你的代码文件中添加log打印一行消息，在手机运行的时候，eclipse会打印代码中出现了log内容，加入某个消息没有执行到，那么说明，在这行代码之前，程序就已经出问题了。

5.导出带签名的apk create key store。。。

**第二种** ： 在终端使用cocos compile -p android -m release 进行打包

说明: -p 代表platform 平台(iOS,android,wp,win,mac) -m release 代表当前发的是代签名的api，需要一个key

1.环境搭建

在.bash\_profile中添加以下几行

export ANDROID\_NDK\_ROOT=/Users/mac/Documents/NDK/android-ndk-r9d

export PATH=$PATH:$ANDROID\_NDK\_ROOT

export ANDROID\_SDK\_ROOT=/Users/mac/Documents/NDK/android-sdk-macosx

export PATH=$PATH:$ANDROID\_SDK\_ROOT

export ANDROID\_ANT\_ROOT=/Users/mac/Documents/NDK/apache-ant-1.9.3/bin

export PATH=$PATH:$ANDROID\_ANT\_ROOT

2.修改Android.mk

3.cd到游戏的工程目录下 执行 cocos compile -p android (debug版本) -m release(发布版本，需要签名)

4.将key文件拖入到终端，根据提示输入相关内容(key的进入密码，alias的密码)

(修改程序包proj.android/jni/Android.mk 13行)

FILE\_LIST := hellocpp/main.cpp

FILE\_LIST += $(wildcard $(LOCAL\_PATH)/../../Classes/\*.cpp)

LOCAL\_SRC\_FILES := $(FILE\_LIST:$(LOCAL\_PATH)/%=%)

cd /Users/student/Desktop/GoldMiners

cocos compile -p android -m release

物理引擎

Box2D

Box2D ：

物理世界 ：(b2world) 是使在世界中的“物体”能够产生 “物理效应” 的一个集合

物理效应：重力 、摩擦 、 冲量 、动能 弹性 、惯性

所谓的世界中的“物体”是指 刚体

刚体：刚体是物理世界的物体，相当于我们cocos2d-x引擎中的精灵，在物理世界中的物体能够受到重力感应

要想能够实现“物体”之间的碰撞 就需要指定 夹具

夹具是物理引擎提出的一个概念 用于能够检测不同“物体”之间的碰撞

quick 终端项目创建：

cd 项目路径

./create\_project.sh -p com.o.m.Test01（最后一个位项目名）

c++与oc的交互

在.mm文件中可以实现C++与Oc的混编，从而实现c++与oc的交互

c++与java交互 –>lua(quick)与java的交互->luaj

lua(quick)与oc的交互 ->luaOC

luaj与luaOC都是放在luaBridge

1. 在quick工程中调用OC的方法
2. OC的方法中会调用在quick工程中指定的回调函数（其实就是C/C++调用lua的函数）在这里我们可以在OC的函数中传递一些变量到quick中

Lua与java交互

包名.类名